Modelado basado en roles con UML

María José Ortín Ibáñez, Jesús García Molina

Departamento de Informática, Lenguajes y Sistemas
Facultad de Informática - Universidad de Murcia
Campus de Espinardo - C.P. 30071 - Murcia
mjortin@dif.um.es, jmolina@fcu.um.es

Resumen: En los últimos años ha aumentado el interés por el modelado orientado a objetos basado en roles. Los modelos de roles permiten representar interacciones entre objetos sin considerar las clases a las que pertenecen hasta llegar a la fase de implementación. Así, la evolución de los métodos parece estar marcada por un abandono de la clase como abstracción principal en favor de las interacciones entre objetos, las cuales son modeladas en el contexto de una colaboración entre objetos que juegan roles. En este trabajo analizamos el concepto de colaboración UML y su correspondencia con los modelos de roles.

1 Introducción

Tradicionalmente, los métodos orientados a objetos han considerado la clase como la principal abstracción en el modelado. Sin embargo, en los últimos años han surgido varios métodos de segunda generación, como *Catalysis* y *OOram* que consideran las interacciones entre objetos como el aspecto esencial del modelado orientado a objetos. Estos métodos incluyen el concepto de *colaboración* como principal mecanismo de abstracción para el modelador, apareciendo las clases como una abstracción relacionada con la implementación.

En un principio, el concepto de *rol* fue introducido en el modelado orientado a objetos para representar cuestiones de modelado avanzado como la clasificación múltiple y la clasificación dinámica. Ahora, sin embargo, el concepto de rol es un elemento central del modelado, ya que es considerado la abstracción adecuada para representar colaboraciones entre objetos, al permitir centrarse en el comportamiento que muestra un objeto en relación al resto de objetos participantes, sin necesidad de determinar la clase a la que pertenece.

El *Lenguaje Unificado de Modelado,* UML, en su versión 1.1, incluyó el concepto de *colaboración* para especificar cómo ciertos elementos, como clases, interfaces y asociaciones, colaboran para ofrecer cierto comportamiento cooperativo mayor que la suma de sus comportamientos individuales. En UML, las colaboraciones se ofrecen como un mecanismo para especificar la realización de casos de uso u operaciones, y para modelar soluciones reutilizables, esto es patrones de diseño y *frameworks*.

En UML las colaboraciones se consideran apropiadas para expresar las construcciones de otros métodos destinadas a modelar interacciones entre objetos, como las incluidas en OOram y Catalysis. La correspondencia entre el mecanismo de

colaboración de Catalysis y el de UML parece bastante directo, en cambio no está tan claro cómo representar modelos de roles de OOram mediante colaboraciones UML.

Aunque la definición de colaboración de UML está basada en el concepto de rol, la literatura sobre UML ha prestado poca atención a la discusión sobre cómo los roles intervienen en las colaboraciones. Por otra parte, en las propias definiciones de UML existe bastante confusión, lo que dificulta considerablemente la comprensión de estos conceptos.

En este trabajo nos proponemos un doble objetivo: i) analizar el concepto de colaboración de UML, subrayando su relación con el concepto de rol, y ii) establecer una correspondencia entre los modelos de roles de OOram y las colaboraciones de UML, mostrando las limitaciones de estas últimas para hacer un verdadero modelado basado en roles.

El trabajo se ha organizado del siguiente modo: en la siguiente sección se define el concepto de rol y se comentan sus aplicaciones más relevantes. En la tercera sección se describe brevemente los modelos de roles OOram. Después se analiza con detalle el concepto de colaboración en UML. En la quinta sección se discute la forma en la que las colaboraciones UML pueden usarse para realizar un modelado con roles, tal y como es propuesto en OOram, presentando a continuación una propuesta y mostrando varios ejemplos de aplicación. Por último se enumeran las conclusiones obtenidas.

2 Roles en el modelado orientado a objetos.

El concepto de rol ha sido introducido en el modelado orientado a objetos con dos propósitos principales: i) representar la clasificación dinámica y clasificación múltiple, y ii) representar colaboraciones entre objetos, habiéndose realizado un gran número de trabajos tanto al nivel de modelado como de implementación. En este trabajo nos centraremos en el segundo de los propósitos.

De acuerdo con el conocido lema “*no object is an island*” expresado en [2], no tiene interés analizar el comportamiento de un objeto aislado, sino que un objeto sólo es interesante en cuanto que colabora con otros objetos para llevar a cabo actividades, y son estas colaboraciones las que interesa describir cuando se modela. Una colaboración representa una interacción entre objetos cuyo propósito es la realización de alguna actividad, y en donde cada objeto participante juega un determinado rol.

Una *clase* describe objetos independientemente del contexto en el que existen, sin tener en cuenta con qué otros objetos interactúan. Por tanto, una clase aislada no es adecuada para describir las actividades en las que participan sus objetos, porque la descripción global de una actividad está repartida entre las clases de los objetos participantes, y por otra parte, si un objeto participa en varias actividades, en su clase se encuentran entremezcladas las descripciones de estas actividades.

En cambio, un *rol* describe las propiedades estructurales y de comportamiento de un objeto en un contexto. Un objeto juega uno o más roles a lo largo de su existencia y un mismo rol puede ser jugado por diferentes objetos. Dado que un objeto finalmente será instancia de una clase, un rol es una proyección (vista parcial) de dicha clase, de modo que las propiedades del rol serán un subconjunto de las propiedades de la clase del objeto. Una clase implementará uno o más roles y un rol podrá ser implementado por una o más clases.

Los métodos de modelado orientado a objetos han evolucionado en el sentido de trasladar el centro de interés de las clases a las colaboraciones entre objetos descritas mediante roles o tipos. Entre los métodos que ofrecen el concepto de colaboración como abstracción básica para el modelado, destacan Catalysis [6] y OOram [14].

El modelado con Catalysis se basa en tres construcciones básicas: *tipo*, *colaboración* y *refinamiento*. Una *colaboración* define un conjunto de acciones entre objetos para proporcionar cierto comportamiento, y cada objeto juega ciertos roles (tipos) en relación con los demás objetos de la colaboración. Un *tipo* especifica el comportamiento visible de un objeto, pero no su implementación. El *refinamiento* es la relación entre una abstracción y su realización, por ejemplo una clase o componente puede ser la realización de cierta interfaz.

OOram es un método centrado en el modelado de las interacciones entre objetos mediante *modelos de roles*. A diferencia de los tipos de Catalysis, un *rol* unifica los conceptos de clase y objeto, ya que describe el comportamiento del objeto y al mismo tiempo posee identidad. En la siguiente sección resumiremos las principales características de los modelos de roles de OOram.

Tanto en OOram como en Catalysis se plantea un modelado orientado a objetos que retrasa la identificación de las clases hasta la fase de implementación, usando los conceptos de rol y tipo, respectivamente. Los roles permiten describir los servicios y características de los objetos sin necesidad de conocer, comprometerse o preocuparse por las clases que los implementarán.

En los últimos años ha aumentado considerablemente el interés por los roles, ya que se consideran la abstracción apropiada para las especificaciones basadas en interfaces, empleadas en el desarrollo de sistemas de objetos distribuidos (como los basados en CORBA), puesto que es más fácil y productivo trabajar con servicios, interfaces y protocolos, evitando concretar qué objetos y clases los soportarán. Así, OOram se ha utilizado como base para definir métodos para una arquitectura basada en objetos distribuidos y el modelo ISO RM ODP [3,9]. En [11] se describe el lenguaje de especificación PSL basado en roles para sistemas distribuidos abiertos. Catalysis [6] también se presenta como un método de modelado para sistemas distribuidos, a partir de componentes y *frameworks*.

UML1.1 [4,5,16] incluye el concepto de colaboración para representar las interacciones entre objetos que juegan roles, con el objetivo de realizar cierta tarea, tal como un caso de uso o una operación. En [4] se indica que las colaboraciones UML son adecuadas para expresar las colaboraciones de Catalysis y los modelos de roles de OOram. Más adelante, describiremos con detalle el concepto de colaboración UML.

3 Modelado con Roles en OOram

OOram (*Object-Oriented Role Analysis and Modeling*) [14,1] es un método de análisis y diseño basado en la metáfora de la orientación a objetos pero que introduce el concepto de *modelo de roles* como principal mecanismo de abstracción que utilizará el modelador. El modelado con roles fue ideado para modelar grandes sistemas y con el propósito de favorecer una implementación en lenguajes de programación orientados a objetos. Para ello el sistema se descompone en un conjunto de subsistemas o áreas de interés que representan actividades desempeñadas por una

estructura de objetos que colaboran entre sí. Cada una de estas estructuras es descrita mediante un modelo de roles.

Un *modelo de roles* describe los objetos que participan en una actividad y las interacciones entre ellos; contiene un conjunto de roles, de modo que todos los objetos que ocupan una misma posición en la estructura son representados por un rol. Un *rol* describe el comportamiento de un objeto en el contexto de una actividad.

La figura 1 muestra el modelo de roles *Compras Proyecto* que describe la actividad asociada a una solicitud de compra por un miembro de un proyecto desarrollado en una organización. El modelo se representa mediante una *vista colaboración* y una *vista escenario*, que son las más utilizadas del conjunto de vistas que ofrece OOram.

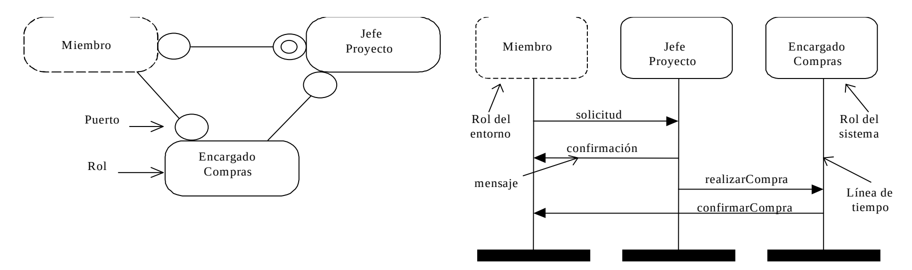


Fig. 1. Modelo de roles *Compras Proyecto*

De la vista escenario se deduce fácilmente la secuencia de acciones que incluye la actividad. Si dos roles están conectados mediante una línea, significa que existe una interacción entre ellos. Si en el extremo de una línea aparece un *puerto*, significa que un objeto que juegue el rol asociado enviará *mensajes* a un objeto que juegue el rol del otro extremo de la línea. El envío de un mensaje provoca la ejecución de una operación o método sobre el objeto del rol que lo recibe. Un puerto representado con un doble círculo denota que la interacción puede ser con un conjunto de objetos.

OOram también incluye la *vista diagrama de estados*, que describe los estados de un rol y las transiciones entre ellos; la *vista proceso* (basada en el estándar IDEF0 [17]) muestra explícitamente las acciones que realizan los roles y los flujos de información que intercambian, y la *vista semántica*, isomorfa a la vista colaboración del mismo modelo, pero que en vez de puertos y caminos de interacción entre roles, muestra relaciones de asociación.

Las propiedades más importantes que caracterizan a los roles de OOram son:

- El concepto de rol unifica los conceptos clase y objeto: los roles tienen tanto una naturaleza estática como dinámica, pues permiten describir las propiedades de los objetos que representan, y también pueden usarse para mostrar cómo los objetos colaboran entre sí.
- Al igual que un objeto, un rol tiene identidad: puede enviar y recibir mensajes.
- La *extensión de un rol* es el conjunto de objetos que pueden representar ese papel; de forma paralela, la *extensión de un modelo de roles* es el conjunto de estructuras de objetos obtenido una vez que hacemos jugar sus roles a los objetos del sistema; durante una instanciación, un rol puede ser jugado por un objeto como máximo.
- Una clase describe un objeto independientemente del contexto en que dicho objeto interacciona con otros; un rol describe un objeto en el contexto de una actividad.

- Los roles son independientes de las clases, permiten un modelado que pospone la elección de las clases que implementan los objetos y de las relaciones entre ellas.
- Un rol puede ser implementado por una o más clases y una clase puede implementar uno o más roles.

4 Roles en UML

UML distingue entre *roles estáticos* y *roles dinámicos*. Los primeros son los roles de extremo de una asociación, utilizados para referirse al papel específico que juega una clase cuando participa en cierta asociación con otra, es decir tienen el mismo significado que en el modelo entidad-relación. Los roles dinámicos son los que participan en las colaboraciones y puesto que, en principio, corresponden a la definición de rol de la sección anterior, vamos a centrarnos en ellos.

El concepto de *colaboración* fue añadido a UML en la versión 1.1 de su semántica [4], con el propósito de describir cómo una operación o clasificador (normalmente un caso de uso) es realizado mediante un conjunto de clasificadores (clases, interfaces, tipos o componentes) y asociaciones entre ellos. Un *clasificador* es un mecanismo UML que describe propiedades estructurales y de comportamiento. Una colaboración define el contexto para definir las interacciones entre instancias de clasificadores. Una *interacción* especifica los mensajes que intercambian un conjunto de instancias para realizar una tarea, y siempre se define en el contexto de una colaboración.

Para describir una colaboración determinada, normalmente no se necesitan todas las características de los clasificadores participantes, ni todas las asociaciones existentes entre ellos. Por esto, una colaboración UML incluye, en lugar de clasificadores y asociaciones, *roles-clasificador* y *roles-asociación*, entendidos como proyecciones de clasificadores y de asociaciones, respectivamente.

Puesto que en este trabajo nos interesa analizar la relación entre roles y clases en UML, hablaremos de *rol-clase* en vez de *rol-clasificador*. Con ello no perdemos generalidad y además es obvio que las clases son los clasificadores que aparecerán la mayor parte de las veces en una colaboración.

Un *rol-clase*, pues, es una vista restringida (proyección) de una clase base, y está definido por las características de la clase que son necesarias para la colaboración. Por tanto, un rol-clase especifica el rol jugado por un participante en la colaboración. Un *rol-asociación* es una proyección de una asociación base y especifica un uso particular de ésta en la colaboración; también es posible encontrar asociaciones entre roles-clase que sólo tienen sentido dentro de cierta colaboración.

Una colaboración no es propietaria de ninguno de sus elementos estructurales, sino que los referencia o usa. Por esto un mismo elemento puede formar parte de más de una colaboración. Una colaboración también puede contener un conjunto de interacciones, usadas para describir el comportamiento de instancias que conforman con los roles-clase participantes. Cuando se instancia una colaboración, a cada rol-clase se liga un objeto (de la clase base), y a cada rol-asociación se liga un enlace (*link*). De este modo toda instancia, objeto o enlace, que participa en una instancia de una colaboración, conforma con un rol, bien un rol-clase o un rol-asociación.

En los documentos que describen las colaboraciones [4,5,16] hemos encontrado bastantes inconsistencias y errores que provocan gran confusión e impidien obtener una perspectiva clara y ordenada. Por ejemplo nos parece que no queda claro por qué las colaboraciones son representadas mediante diagramas de colaboración en los que unas veces aparecen instancias de clases y otras veces roles-clase. Por otra parte, no aparece una notación que permita representar en un mismo diagrama varios objetos que jueguen el mismo rol. Además, no se indica cómo definir los roles-clase y cómo asociarlos a sus clases base.

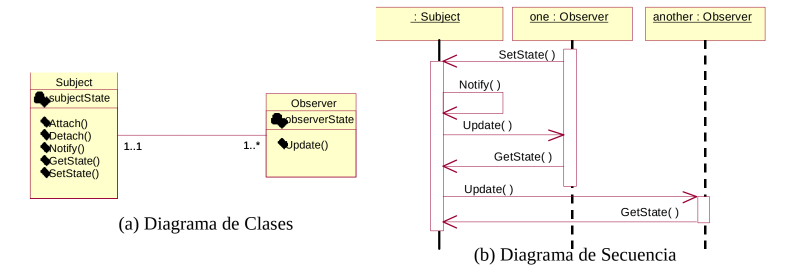
A continuación exponemos nuestra visión sobre cómo utilizar las colaboraciones UML, ofreciendo una interpretación de los conceptos relacionados e indicando cómo debería ser la notación.

- Una colaboración puede ser presentada a dos niveles: de especificación y de instancia. Una *colaboración al nivel de especificación* contiene roles-clase conectados por roles-asociación, y representa una interacción potencial entre los roles-clase; una colaboración en este nivel puede ser instanciada muchas veces para producir instancias de colaboración. Una *colaboración al nivel de instancia* muestra una interacción concreta entre objetos que juegan roles y que están conectados mediante enlaces.
- Una colaboración tiene dos aspectos: el *aspecto estructural* o *estático* consiste en un conjunto de roles y las relaciones estructurales entre ellos; el *aspecto dinámico* o *de comportamiento* consiste en una o más interacciones entre los roles participantes en la colaboración.
- El aspecto estructural de una colaboración puede ser expresado en un diagrama de clases. Según su definición, este diagrma puede contener cualquier tipo de clasificador, por lo que puede incluir roles-clase y así definir el subconjunto de características de la clase base que incorpora cada rol-clase. Este diagrama siempre muestra la colaboración al nivel de especificación.
- El aspecto de comportamiento puede expresarse en diagramas de colaboración o secuencia, los cuales pueden estar al nivel de especificación o al nivel de instancia.
- Un *diagrama de colaboración al nivel de especificación* contiene roles-clase conectados por roles-asociación. Es una descripción genérica de una interacción entre objetos que juegan roles, que puede contener bucles, bifurcaciones, etc. Un rol-clase se representa igual que una clase y su nombre tiene el formato *nombreRol:NombreClaseBase*; por ejemplo el rol-clase *profesor:Persona* denota el rol *profesor* para la clase base *Persona*. Es posible mostrar la multiplicidad de un rol-clase (número de objetos que pueden jugar a la vez dicho rol en una instancia de la colaboración). Un rol-asociación puede tener un nombre con formato *nombreAsociacion:NombreAsociacionBase*, aunque puede omitirse el nombre de la asociación base si ésta no existe fuera de la colaboración. Además puede incluir adornos usuales para asociaciones (multiplicidad, flecha de navegabilidad, etc).
- Un *diagrama de colaboración al nivel de instancia* expresa una interacción particular, sin bifurcaciones ni bucles, entre objetos que intercambian mensajes a través de enlaces. Los objetos y enlaces participantes conforman con roles-clase y roles-asociación, respectivamente, de la colaboración. El nombre de cada objeto revela el rol que juega: *nombreRol:NombreClaseBase*. El nombre de la clase base puede omitirse si se sobreentiende en el contexto. Con *profesor:Persona* o *profesor* se denota un *objeto cualquiera* de la clase *Persona* que juega el rol *profesor*. Con *:Persona* se representa un objeto que no juega un rol específico, sino que su

comportamiento en la colaboración está definido por su clase, lo que sucedería si todas las personas consideradas en el sistema fueran profesores y sólo se comportaran como tales; podría utilizarse <u>*p:Persona*</u>, donde *p* es un nombre de instancia y no de rol, para distinguirlo de otros objetos *Persona*. Por último, <u>*Elena:Persona*</u> sería el nombre de un *objeto existente* en el mundo real.

- Un *diagrama de secuencia* muestra una interacción entre objetos que juegan roles.
- Un diagrama de colaboración sin mensajes muestra el *contexto* en el que pueden ocurrir las interacciones entre los participantes, sin mostrar ninguna de ellas; si contiene mensajes, representa *una interacción*, de forma análoga a como lo hace un diagrama de secuencia.

Es importante destacar que en cualquier caso, siempre hemos de conocer la clase sobre la que se define el rol-clase, o de la que el objeto es una instancia cualquiera, excepto en el caso de que se use una colaboración parametrizada.



(a) Diagrama de Clases

(b) Diagrama de Secuencia

Fig. 2. El patrón de diseño *Observer*

UML utiliza colaboraciones parametrizadas para modelar la estructura de patrones de diseño [9] (ver figura 2). Una *colaboración parametrizada* representa una colaboración genérica, en la que algunos participantes son parámetros. Normalmente, los parámetros son clases base de roles-clase (*Subject* y *Observer* en el ejemplo). Cuando se instancia una colaboración parametrizada, ligando elementos del modelo (clases) a los parámetros, se obtiene una colaboración concreta. En el interior de la colaboración, el aspecto estructural del patrón de diseño puede ser representado en un diagrama de clases en el que se muestran los roles componentes del patrón (fig. 2.a), y el aspecto de comportamiento puede ser representado en diagramas de interacción (fig. 2.b). En el exterior, puede verse cómo se aplica el patrón, ligando los parámetros de la colaboración parametrizada con abstracciones (clases) existentes en el sistema.

5 Propuesta para un modelado con roles en UML

En la construcción de sistemas tales como las aplicaciones de objetos distribuidos, es esencial una técnica de modelado basado en roles al estilo OOram, que permita trabajar con roles de objetos, sin preocuparse por las clases que implementarán los objetos. La identificación de las clases y la asignación de los roles a las clases que los implementan, se deja para posteriores etapas en el diseño del sistema.

Antes de la inclusión del concepto de colaboración en UML, T.Reenskaug presentó en [12,13] una propuesta que contempla la existencia de roles sin clase base asociada,

a los que es posible asignar atributos y operaciones independientemente de las clases que los implementen. Sin embargo, esta idea no fue incluida en la versión UML1.1 definitiva, sino que, como hemos analizado en la sección anterior, la introducción de un nuevo rol-clase requiere la existencia de una clase base, y toda instancia denota la manifestación concreta de una clase. En esta sección analizaremos si a pesar de tal restricción es posible todavía realizar en UML un modelado basado en roles que respete el espíritu *"roles antes que clases"*. Para ello, primero introduciremos varias alternativas y a continuación describiremos nuestra propuesta, que aprovecha los mecanismos de extensibilidad de UML.

Catalysis, que utiliza como notación UML, resuelve el problema definiendo un tipo para cada rol. En los diagramas de interacción, un nombre de objeto tiene el formato *nombreObjeto:nombreTipo*. La determinación de las clases del sistema y la asignación de roles a clases, se retrasan hasta la etapa de *Diseño del Sistema*, en la cual, si las instancias de cierta clase pueden jugar un rol, se especifica que la clase implementa el tipo correspondiente. Este enfoque plantea varios problemas: i) no está clara la relación entre los tipos y los roles-clase de las colaboraciones UML, y ii) un tipo UML es un estereotipo de clase cuya semántica corresponde a la de un tipo de datos: especificación de la estructura y comportamiento de un conjunto de objetos (dominio del tipo).

Otra alternativa relacionada con la solución que aporta Catalysis, sería usar interfaces en vez de tipos, pero tampoco deja clara la relación entre interfaces y roles-clase y además, las interfaces no pueden tener atributos, necesarios para representar el estado de los roles.

De acuerdo con lo expuesto en la sección anterior, un rol-clase es definido a partir de una clase base, excepto en el caso de las colaboraciones parametrizadas, en las que la clase base es un parámetro de la plantilla. Por tanto este tipo de colaboración podría ser una solución: primero crear el modelo de roles y luego una vez identificadas las clases, utilizar éstas para instanciar el patrón. Sin embargo, esta solución no nos parece adecuada, ya que las colaboraciones parametrizadas están destinadas a encapsular un patrón que aparece de forma recurrente (una solución común a un problema común). Es un modelo definido y almacenado para su aplicación en diversos contextos, lo cual constituye un caso particular de modelo de roles.

Detallamos ahora nuestra propuesta para trasladar a UML el modelado con roles de OOram. Un rol OOram posee tanto un aspecto estructural (clase) como dinámico (objeto); sin embargo UML está basado en la *dicotomía clase/objeto*, por lo cual hemos de considerar por separado los dos aspectos de un rol. Así, proponemos la definición de un estereotipo de clase llamado «role», que corresponde a la parte descriptiva de un rol OOram, y utilizar este elemento como clase base de los roles-clase de las colaboraciones. En realidad, un rol-clase y su «role» asociado *son una misma cosa*, de modo que el modelador sólo pensará en términos de roles. Para nombrar un rol-clase se usará *:nombreRol* (nivel de especificación), o *:nombreRol* (nivel de instancia). Por ejemplo, si definimos el «role» *Profesor*, el rol-clase se denominará *:Profesor*; podría ir precedido de un nombre para distinguir entre varios objetos que jueguen un mismo rol.

El modelado de cualquier aspecto de una colaboración estará basado en roles con la misma semántica que OOram, sólo que la descripción de cada rol se realizará mediante un elemento «role», y no será necesario considerar clases.

Un esquema de la correspondencia entre los conceptos de OOram y de UML sería:

1. Por cada área de interés OOram (actividad, proceso de negocio) se definirá un *caso de uso* UML *de granularidad alta*, que será realizado a través de una *colaboración*.
2. Se debe identificar los roles que juegan los objetos dentro del contexto de cada colaboración, y crear para cada rol un elemento «role», que puede incluir algunas operaciones y/o atributos predefinidos, y una lista de responsabilidades.
3. Se debe analizar cómo los roles colaboran entre sí (las posibles interacciones) para llevar a cabo la correspondiente actividad. Crear un *diagrama de secuencia* que visualice el escenario de *cada* interacción. Después se puede crear el correspondiente *diagrama de colaboración*, que muestra de forma explícita las relaciones entre los participantes. El conjunto de diagramas de colaboración y secuencia describirá el *aspecto dinámico* de la colaboración.
4. Una vez se conoce con qué roles interactuará cada rol participante en la colaboración, se podrá construir un *diagrama de roles*, formado por los elementos «role» identificados, conectados por asociaciones según las relaciones estructurales entre ellos. Este diagrama mostrará el *aspecto estático* de la colaboración.
5. El *contexto de las interacciones* se visualiza en un *diagrama de colaboración sin mensajes al nivel de especificación*. Conviene destacar que, a diferencia del diagrama de roles, en este diagrama no es posible representar la estructura interna de cada rol, ni mostrar relaciones entre roles plenamente estructurales.
6. En este punto, el comportamiento de cada rol se conoce con bastante exactitud: el rol-clase incluirá un conjunto de operaciones obtenido a partir de los mensajes recibidos por el mismo, en los diferentes diagramas de interacción.
7. Finalmente, podrá crearse un diagrama de clases inicial, a partir del diagrama de roles, en el que cada clase será la implementación de uno o más roles.

A continuación indicamos la correspondencia entre las vistas de OOram y los diagramas de UML. Subrayamos la necesidad de definir los elementos «role» para especificar las características de cada rol, y así poder crear estos diagramas.

- Una vista colaboración OOram corresponde a un diagrama de colaboración UML sin mensajes al nivel de especificación. Cada rol está representado por un rol-clase, y los puertos por la multiplicidad y flecha de navegabilidad de los roles-asociación.
- Una vista escenario corresponde a un diagrama de secuencia UML con instancias que conforman con roles-clase.
- Una vista proceso se representa mediante un diagrama de actividades UML en el que cada *swimlane* se asigna a un rol.
- Una vista diagrama de estado corresponde a un diagrama de estados UML.
- Una vista semántica corresponde a un diagrama de roles UML. En OOram esta vista se usa muy poco, pues no suele aportar nueva semántica al modelo. Sin embargo en UML, los diagramas de roles son imprescindibles para evitar identificar desde un principio las clases sobre las que definir los roles.

6. Ejemplos de aplicación

Realización de un proceso de negocio

Para el proceso de negocio de la figura 3, correspondiente al ejemplo OOram de la figura 1, se ha definido los «role» *Miembro*, *Jefe Proyecto* y *Encargado Compras*. El primer diagrama describe la interacción entre los *roles* implicados. El contexto de la colaboración se expresa mediante un diagrama de colaboración sin mensajes, en el que es posible expresar la multiplicidad de los roles-clase y la navegabilidad de los caminos de interacción representados por roles-asociación.

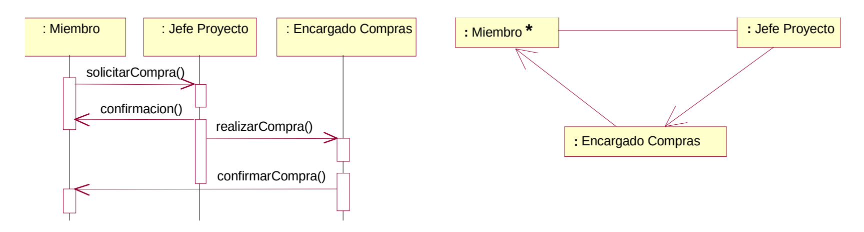


Fig. 3. Realización del proceso de negocio *Compras Proyecto*

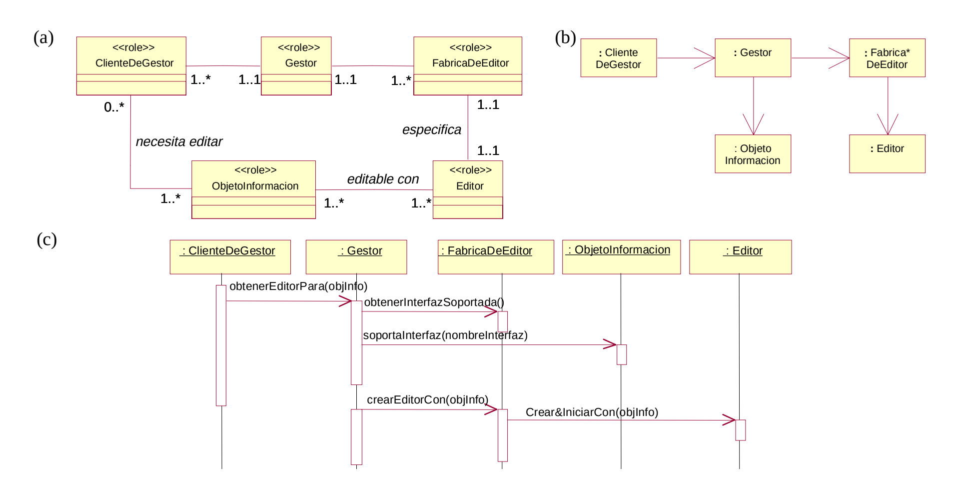


Fig. 4. Realización de la operación *Editar Objetos*

Realización de una operación

El objetivo del mecanismo *Editar Objetos* (fig. 4) es seleccionar un tipo de editor que pueda editar un objeto de información, instanciar el editor y ligarlo a dicho objeto. Un tipo de editor (AcrobatReader,Word...) puede editar objetos de información diferentes (documentos word, postscript, html..) y un objeto puede ser editado por distintos

editores. El diagrama de roles de la figura 4.a muestra el aspecto estructural de la colaboración. El diagrama de colaboración sin mensajes de la figura 4.b refleja el contexto de la interacción entre los roles. La figura 4.c muestra el diagrama de secuencia que expresa una posible interacción entre los objetos (roles).

7 Conclusiones y trabajo futuro

En este trabajo hemos analizado las colaboraciones de UML y el grado en que permiten desarrollar un modelado basado en roles, en el cual las clases son consideradas un concepto ligado a la implementación. Como resultado del trabajo realizado podemos establecer las siguientes conclusiones:

- En el presente trabajo se ofrece una interpretación precisa de los conceptos relacionados con las colaboraciones de UML1.1, y la notación asociada, que hasta ahora no habían sido tratados con la suficiente claridad en la literatura.
- A pesar de que UML1.1 define el concepto de colaboración a partir del concepto de rol, no permite un modelado independiente de las clases, ya que cada rol de una colaboración se define como una proyección de una clase existente, que especifica el aspecto de la clase que interesa para la colaboración. Por otro lado, un rol-clase no tiene una identidad que le permita comportarse como un objeto.
- Es posible usar colaboraciones parametrizadas para representar patrones de diseño, pero no es apropiado usar este mecanismo para representar modelos de roles en general. Por otra parte, el enfoque de Catalysis de emplear tipos UML como roles, no hace uso de los roles de las colaboraciones de UML.
- Creemos que la propuesta de utilizar elementos «role» como base de los roles-clase, conceptos que se consideran equivalentes, permite un modelado *"roles antes que clases"*. Hemos utilizado el nuevo concepto para establecer una correspondencia entre las vistas de OOram y los diagramas UML.
- UML está en continua evolución. T. Reenskaug comenta que dicha evolución se está produciendo en el sentido de abandonar la visión "tradicionalista" basada en clases, en favor de una visión en la que roles y clases están al mismo nivel [15]. Recientemente, en la fase final del presente trabajo, ha salido a la luz la versión 1.3 beta R1 de UML (*http://uml.shl.com/artifacts.htm*). Después de analizar los cambios introducidos, que solventan algunas de las ambigüedades y errores antes señaladas, hemos comprobado que se ajustan a la interpretación dada en la sección 4.. Sin embargo, aun teniendo en cuenta estos cambios, no es posible todavía poner en práctica la máxima de "*roles antes que clases*", puesto que es necesario definir las clases para definir los roles-clase.
- Como establece D. Lea en [10] "*es difícil predecir cuándo se necesitará la flexibilidad y potencia proporcionada por los roles, de modo que un enfoque conservador es comenzar siempre con roles*". En esta línea, entendemos que UML debería permitir un modelado con roles sin considerar las clases hasta el final.

Tanto OOram como UML incluyen mecanismos para crear nuevos modelos a partir de otros existentes (*síntesis* en OOram y *composición de colaboraciones* en UML), utilizados con fines de reutilización, por ejemplo. Ambos conceptos parecen estar muy próximos y trataremos su correspondencia en posteriores trabajos.

Referencias

1. Andersen, E.P.: Conceptual Modeling of Objects: A Role Modeling Approach. PDThesis. Faculty of Mathematics and Natural Sciences. University of Oslo. Norway. 1997.
2. Beck, K., Cunningham, W.: A Laboratory for Teaching Object-Oriented Thinking. Procs. of OOPSLA'89, SIGPLAN Notices, vol 24, no.10. October 1989.
3. Berre, A.J., Aagedal, J.O., Silva, A.R.: SIMOD - An ODP-extended Role Modeling Methodology for Distributed Objects. Procs. of HICSS-30. Wailea.Hawai.USA.1997.
4. Booch, G. et al.: UML Notation Guide and UML Semantics version 1.1. September 1997. http://www.rational.com/uml/index.jtmpl.
5. Booch, G., Rumbaugh, J., Jacobson, I.: The Unified Modeling Language User Guide. Addison-Wesley. 1999.
6. D'Souza, D.F., Wills, A.C.: Objects, Components and Frameworks with UML: The Catalysis Approach. Addison-Wesley. 1997.
7. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley. 1995.
8. García Molina, J., Acacio, M.E., García, G.: Una experiencia práctica con la Arquitectura de Tres Modelos de OOram. Actas del IV Congreso sobre Tecnología de Objetos. U. Deusto. Bilbao. 1998.
9. Henning, T.: MODS: A role-based Method for analysis and desing of Object-Oriented Distributed Systems.Diploma Thesis.University of Technology and Science.Norway.1997.
10. Lea, D.: Roles Before Objects. http://g.oswego.edu/dl/rp/roles.html. 1995.
11. Lea, D., Marlow, J.: PSL: Protocols and Pragmatics for Open Systems. ftp://g.oswego.edu/pub/papers/psl.ps.
12. Oldevik, J.; Berre, A.J., Reenskaug, T.: Formalising the notion of object identity in UML Version 0.1. Technical Report. Taskon. Norway. April 1997.
13. Reenskaug, T.: Merging Role Models with Collaborations and Use Cases. Technical Report. Taskon. Norway. May 1997.
14. Reenskaug, T.: Working with Objects: the OOram software engineering method. Addison-Wesley/Manning Publications. 1996.
15. Reenskaug, T. Comunicación personal vía correo electrónico.
16. Rumbaugh, J., Jacobson, I., Booch, G.: The Unified Modeling Language Reference Manual. Addison-Wesley. 1999.
17. Software Standard Integration Definition form Function Modeling (IDEF0). Federal Information Processing Standards Publications 183, 1993.

Agradecimientos

Deseamos expresar nuestro agradecimiento a Trygve Reenskaug por sus comentarios, que han sido de gran utilidad durante la elaboración del presente trabajo.